Local scene flow by tracking in intensity and depth

Julian Quiroga, Frédéric Devernay, James L. Crowley

Local Scene Flow by Tracking in Intensity and Depth

Julian Quiroga[a,b,*], Frédéric Devernay[a], James Crowley[a]

[a]*INRIA Grenoble Rhone-Alpes, 655 avenue de l'Europe, 38334 Saint Ismier Cedex, France*
[b]*Departamento de Electrónica, Pontificia Universidad Javeriana, Bogotá, Colombia*

Abstract

The scene flow describes the motion of each 3D point between two time steps. With the arrival of new depth sensors, as the Microsoft Kinect, it is now possible to compute scene flow with a single camera, with promising repercussion in a wide range of computer vision scenarios. We propose a novel method to compute a local scene flow by tracking in a Lucas-Kanade framework. Scene flow is estimated using a pair of aligned intensity and depth images but rather than computing a dense scene flow as in most previous methods, we get a set of 3D motion vectors by tracking surface patches. Assuming a 3D local rigidity of the scene, we propose a rigid translation flow model that allows solving directly for the scene flow by constraining the 3D motion field both in intensity and depth data. In our experimentation we achieve very encouraging results. Since this approach solves simultaneously for the 2D tracking and for the scene flow, it can be used for motion analysis in existing 2D tracking based methods or to define scene flow descriptors.

Keywords: Scene flow, 3D motion estimation, Image tracking, Brightness consistency, Depth data, Locally-rigid motion, Optical flow, Image warping

1. Introduction

The scene flow corresponds to the 3D motion field of the scene [1] and since it provides the motion of 3D points, an accurate estimation of the scene flow can be useful in a wide variety of applications including navigation, interaction, object segmentation, motion analysis, tracking, *etc*.

A current topic of great interest is human activity understanding, where video analyzing and interpretation is required to perform recognition or classification, and the 3D information given by the scene flow may be used to provide powerful features. However, there is no work that directly computes scene flow to perform tasks like human action recognition or gesture classification. Probably, this is due to the fact that most existing methods require stereo or multi-view camera systems, which are not always available. Besides, most of these methods compute a dense scene flow by optimizing a global energy function, spending a lot of processing time and becoming not suitable for real time applications. On the other hand, the optical flow that is related with the scene flow projection on the image plane has been successfully used in human action recognition. Histograms of optical flow are commonly used in state-of-the-art techniques in action recognition to construct descriptors over spatio-temporal interest points [2, 3, 4] and to extract 2D trajectories by tracking key-points [4, 5, 6]. Furthermore, since trajectory based methods outperform other state-of-the-art approaches for action classification [7], it is promising to use scene flow to capture motion information by extracting accurate 3D trajectories.

Recently with the arrival of depth cameras, based either on time-of-flight (ToF) or structured light sensing, it has been possible to compute scene flow using a pair of registered sequences of depth and intensity, as recently in [8, 9]. Depth sensors have decreased system requirements needed to compute the 3D motion field, opening the door to incorporate scene flow based features in common recognition tasks. Some attempts have been made to include depth data in human action recognition tasks. For example, a bag of 3D points extracted from the depth data is used for recognition in [10] while in [11] descriptors obtained by well known techniques [2, 12] were extended with depth information, outperforming the original methods. Similarly, when a depth sensor is available the scene flow can be inferred from the optical flow by using the depth information. However, as we show in this paper, even small errors in the optical flow may generate significant errors in the scene flow computation. Computing scene flow in this way does not fully exploit the relation between the intensity and depth information. In this work we aim to explore how to simultaneously use intensity and depth data to compute local scene flow. As a result, we propose a method that can be used to get accurate 3D trajectories and define scene flow based descriptors.

One of the main contributions of this paper is the definition of a pixel motion model that allows the constraint of the scene flow in the image. Using this motion model and assuming a 3D local rigidity of the scene, we are able to solve for the scene flow that best explains the observed intensity and depth data. Therefore, our method combines information of both sensors and simultaneously solve for the scene flow and its projec-

*Corresponding author

Email addresses: julian.quiroga@inria.fr (Julian Quiroga), frederic.devernay@inria.fr (Frédéric Devernay), james.crowley@inria.fr (James Crowley)

tion in the image, which we named *image flow*. This approach differs from previous scene flow methods using depth sensors, since they reconstruct the scene flow from the observed optical flow [8] or by using a large number of hypotheses to explain the motion of each point in 3D [9], without exploiting the 2D parameterization. Moreover, unlike other scene flow methods that suffer from the smoothness constraint brought by 2D parameterization, we use a 3D local rigidity assumption, which is approximately real for most of the scenes of interest.

The other contribution of the paper is the formulation of a local scene flow computation method by extending the Lucas-Kanade framework [13] to exploit both intensity and depth data. In this formulation, it is possible to treat with large displacements in a coarse-to-fine procedure. Besides, instead of solving for a dense scene flow by optimizing a global energy, as most of previous methods, we solve for a local scene flow that can be focused over a selected set of key-points. This formulation is versatile enough to extract accurate 3D trajectories, initialize other methods by computing a dense scene flow, or refine a estimated 3D motion field over specific points. Unlike previous scene flow methods our local approach is suitable for real time applications and its extension to multiple cameras or depth sensors is straightforward.

1.1. Related work

Scene flow was first introduced by Vedula *et al.* [1] as the full 3D motion field in the scene. Most scene flow methods assume a stereo, or multi-view camera system, in which the motion and the geometry of the scene are jointly estimated, in some cases, under a known scene structure. Since optical flow is (an approximation of) the projection of the 3D motion field on the camera image plane, an intuitive way to compute scene flow is to reconstruct it from the optical flow measured in a multi-view camera system, as proposed by Vedula *et al.* [14], or including a simultaneously structure estimation as Zhang and Kambhamettu [15]. However, it is difficult to recover a scene flow compatible with several observed optical flows that may be contradictory.

The most common approach for estimating scene flow is to perform an optimization on a global energy function, including photometric constraints and some regularization. Some authors introduce constraints of a full calibrated stereo structure [16, 17, 18, 19, 20]. Wedel *et al.* [17] enforce consistency on the stereo and motion solution but they decouple the disparity at the first time step without exploiting the spatio-temporal information. To overcome this limitation, simultaneous solution of the scene flow and structure was proposed. Huguet and Devernay [18] simultaneously compute the optical flow field and two disparities maps, while Valgaerts *et al.* [21] assume that only the camera intrinsics are known and they show that scene flow and the stereo structure can be simultaneously estimated.

All these methods suffer from the smoothness constraints brought by 2D parametrization. Basha *et al.* [19] improve the estimation by formulating the problem as a point cloud in 3D space and the scene flow is regularized using total variation (TV). Recently, Vogel *et al.* [20] regularize the problem by encouraging a locally rigid 3D motion field, outperforming TV regularization. Furthermore, other methods simultaneously solve the 3D surface and motion [22, 23]. Another possibility is to work in the scene domain, and to track 3D points or surface elements [24, 25]. Carceroni and Kutulakos [24] model the scene as a set of surfels but it requires a well-controlled lighting and acquisition setup, and because its complexity the scene flow solution is only suitable in a limited volume. Rather than computing a dense scene flow, Devernay *et al.* [24] directly get a set of 3D trajectories from which the scene flow is derived. However, this method suffers from drifts problems and its proposed point visibility handling is a difficult task.

When a depth camera is available, the sensor provides structure information and surface estimation is not needed. Spies *et al.* [26] estimate the scene flow by solving for the optical flow and range flow. Lukins and Fisher [27] extend this approach to multiple color channels and one aligned depth image. In these approaches the 3D motion field is computed by constraining the flow in intensity and depth images of an orthographically captured surface, so that, the range flow is not used to support the optical flow computation. Letouzey *et al.* [8] directly estimate the 3D motion field using photometric constraints and a global regularization term without fully exploiting the information given by the depth sensor. Recently, Hadfield and Bowden [9] estimate the scene flow by modeling moving points in 3D using a particle filter, reducing the over-smoothing caused by global regularization. However, this method requires a lot of computational time since a large number of motion hypotheses must be generated and tested for each 3D point.

1.2. Our approach

Similar to [8, 9], we estimate the scene flow using a pair of aligned intensity and depth sequences. Although, rather than computing a dense scene flow we get a set of 3D motions by tracking in the image domain using a coarse-to-fine procedure. The work in this paper is inspired by that of Devernay *et al.* [25], in which a sparse scene flow is derived from 3D trajectories using several cameras. In our approach, instead of tracking 3D points we use a Lucas-Kanade framework [13] to solve for a local scene flow using constraints in intensity and depth data.

Previous works [26, 27] solve at the same time for the optical flow and range flow assuming an orthographic camera, however, under this approach the estimated depth velocity can not be included to constraint the optical flow computation. Instead, we directly compute a local scene flow by tracking small surface patches in intensity and depth data. As in [20], we assume that the scene is composed of independently, but rigidly, moving 3D parts avoiding the use of smoothness constraints in 2D. Thus, considering a 3D local rigidity of the scene, we model the image flow induced by the surface motion by using a *rigid translation flow model*. This model allows the constraint of the 3D motion field in the image domain. In this way we are able to solve for the scene flow that best explains the observed intensity and depth data for each interest region on the image.

Previous Lucas-Kanade methods use a 2D warping [13]. Unlike them, we model the image flow as a function of the 3D motion vector with help from a depth sensor, improving the accuracy of the optical flow and solving directly for the scene

flow. Besides, without expecting a planar surface patch as in surfels based techniques [24, 25], this motion model allows the constraint of scene flow in intensity and depth images. In order to treat with large displacement the scene flow can be propagated in a coarse-to-fine strategy.

Incorporating depth data our approach improves tracking precision in the image domain that allows at the same time the computation of the scene flow. Because we solve directly for the scene flow by performing tracking in the image domain, this method can be directly used to extract accurate 2D or 3D trajectories, initialize/refine other scene flow methods or define scene flow based descriptors for motion analysis.

1.3. Paper structure

The remainder of this paper is organized as follows. We begin in Section 2 with the definition of a motion model that allows the constraint of the 3D motion vector in the image domain. In Section 3 we describe how this motion model can be used, in a Lucas-Kanade framework, to solve for the local scene flow using constraints from the intensity data. In Section 4, we extend this formulation to include constraints from depth data. In Section 5 we present the experimental results and we discuss the major conclusions that can be derived from them.

2. Pixel motion model

In order to constrain the scene flow in the image domain, it is required a model to explain the image flow induced by 3D motions in the scene. We consider a fixed camera observing a scene composed by locally-rigid 3D parts.

The instantaneous motion, relative to the camera, of a rigid surface in the scene can be decomposed into two components: a translation velocity $\mathbf{V} = (V_X, V_Y, V_Z)$ and an angular velocity $\mathbf{W} = (\Omega_X, \Omega_Y, \Omega_Z)$. The camera frame is defined with its origin at the optical center and its (X, Y, Z) axes respectively in the direction of the image axes and the optical axis, with positive Z pointing in the direction of sight. Let $\mathbf{X} = (X, Y, Z)$ be coordinates at time $t - 1$ of a surface point in the camera reference frame and let $\mathbf{X}' = (X', Y', Z')$ be the corresponding coordinates at time t. If between $t - 1$ and t the surface performs a rotation $\mathbf{W}$ followed by a translation induced by velocity $\mathbf{V}$ then

$$\mathbf{X}' = \mathbf{R}\mathbf{X} + \mathbf{V}, \tag{1}$$

where the rotation matrix $\mathbf{R}$ can be approximated (assuming small values of $\mathbf{W}$ [28]) by

$$\mathbf{R} = \begin{pmatrix} 1 & -\Omega_Z & \Omega_Y \\ \Omega_Z & 1 & -\Omega_X \\ -\Omega_Y & \Omega_X & 1 \end{pmatrix}. \tag{2}$$

2.1. Rigid translation flow model

Let $\mathbf{x} = (x, y)$ be the projection of a surface point $\mathbf{X}$ on the image plane, if the camera matrix $\mathbf{M}$ is given by

$$\mathbf{M} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}, \tag{3}$$

then $x = (Xf_x/Z) + c_x$ and $y = (Yf_y/Z) + c_y$ (we suppose in the rest of this paper that nonlinear distortion was removed from the images). In a similar way the image coordinates $\mathbf{x}' = (x', y') = \hat{\mathbf{M}}(\mathbf{X}')$ is the projection of the point $\mathbf{X}'$, where $\hat{\mathbf{M}}$ is the projective function derived from (3). The 3D movement of the surface generates a motion of its projection. Let (u, v) be the *image flow* (an element of the 2D motion field) induced on pixel $\mathbf{x}$ by rotation and translation of the surface, then

$$u = x' - x = f_x\left(\frac{X - Y\Omega_Z + Z\Omega_Y + V_X}{Z - X\Omega_Y + Y\Omega_X + V_Z} - \frac{X}{Z}\right) \tag{4}$$

and

$$v = y' - y = f_y\left(\frac{Y - X\Omega_Z + Z\Omega_X + V_Y}{Z - X\Omega_Y + Y\Omega_X + V_Z} - \frac{Y}{Z}\right). \tag{5}$$

The image flow induced by the 3D motion does not necessarily correspond with the optical flow, which is defined as the apparent motion of brightness patterns in the image.

We define the *Rigid Translation Flow Model* assuming that *i*) the contribution of the inter-frame rotation is negligible and *ii*) the Z component of the translational velocity is very small w.r.t. Z. Starting from (4) and (5), we first neglect the rotation term $\mathbf{W}$ so that the image flow can be written as:

$$u = f_x\left(\frac{X + V_X}{Z + V_Z} - \frac{X}{Z}\right) = \frac{1}{Z}\left(\frac{f_x V_X - (x - c_x)V_Z}{1 + V_Z/Z}\right) \tag{6}$$

and

$$v = f_y\left(\frac{Y + V_Y}{Z + V_Z} - \frac{Y}{Z}\right) = \frac{1}{Z}\left(\frac{f_y V_Y - (y - c_y)V_Z}{1 + V_Z/Z}\right). \tag{7}$$

Now, assuming that $|V_Z/Z| \ll 1$, the image flow induced on a pixel (x, y) by the surface translation is modeled by

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{Z}\begin{pmatrix} 1 & 0 & -x \\ 0 & 1 & -y \end{pmatrix}\begin{pmatrix} V_X \\ V_Y \\ V_Z \end{pmatrix}, \tag{8}$$

where, for brevity and without loss of generality, we have considered the focal lengths as the unit and the optical center at the image origin. In general, the motion of each pixel can be expressed as a function of its depth Z, its current image position $\mathbf{x}$, the surface motion $\mathbf{V}$ and the intrinsic camera matrix $\mathbf{M}$. The resulting image flow in expression (8) agrees with that of Longuet [29] for the translational component of the instantaneous velocity of a retinal image point.

The rigid translation model (8) can be used to describe the flow induced on the image for all visible points of a rigid surface performing a 3D translation $\mathbf{V}$.

2.2. Other motion models

The inter-frame image flow induced by the 3D motion of a rigid surface has been modeled in the literature using different assumptions. The instantaneous velocity of a retinal point (x, y) is modeled in [29] as

$$u = \frac{1}{Z}(V_X - xV_Z) - xy\Omega_X + \left(1 + x^2\right)\Omega_Y - y\Omega_Z \tag{9}$$

$$v = \frac{1}{Z}(V_Y - yV_Z) - \left(1 + y^2\right)\Omega_X + xy\Omega_Y + y\Omega_Z \tag{10}$$

with $(u, v) = (dx/dt, dy/dt)$, and (V_X, V_Y, V_Z) and $(\Omega_X, \Omega_Y, \Omega_Z)$ the instantaneous translation vector and angular velocities, respectively. Horn *et al.* [30] use this instantaneous velocity model to describe the inter-frame pixel motion, where velocities become displacements between $t-1$ and t, and the optical flow is defined as $(u, v) = (x' - x, y' - y)$. Adiv [31] shows that equations (9) and (10) can be used to approximate the image flow defined by (4) and (5) under the assumptions that the view of field of the camera is not very large, and V_Z/Z ratio and rotation velocities are very small. This model is named *Rigid Body Model* in the hierarchy classification of motions proposed by Bergen *et al.* [32], and it is characterized by 6 degrees of freedom which are functions of the 3D motion and the depth Z.

If the optical flow corresponds to the projection of a planar surface $k_X X + k_Y Y + k_Z Z = 1$, the plane parameters (k_X, k_Y, k_Z) constrain the flow induced by the 3D motion of the surface. This optical flow model is called *Planar Surface Flow* [32] and it is expressed as a function of 8 parameters in [31] as

$$\begin{aligned} u &= a_1 + a_2 x + a_3 y + a_7 x^2 + a_8 xy \\ v &= a_4 + a_5 x + a_6 y + a_7 xy + a_8 y^2, \end{aligned} \tag{11}$$

where coefficients $\{a_1, ..., a_8\}$ are functions of the motion parameters $\{(V_X, V_Y, V_Z), (\Omega_X, \Omega_Y, \Omega_Z)\}$ and the surface parameters (k_X, k_Y, k_Z). This 8 DOF model is also known as the *Homography Model* in [13].

In [32] the surface is considered to be far from the camera and rotation components Ω_x and Ω_y are assumed to be zero, which leads to the *Affine Flow Model*, defined as

$$\begin{aligned} u &= a_1 + a_2 x + a_3 y \\ v &= a_4 + a_5 x + a_6 y, \end{aligned} \tag{12}$$

In this case the induced flow is modeled by the coefficients $\{a_1, ..., a_6\}$ which are functions of the depth Z, the motion parameters (V_X, V_Y, V_Z) and the rotational component Ω_Z. Finally, the simplest model is defined assuming negligible angular velocities and an orthographic camera model, in which Z displacements do not affect the projection of the 3D point on the image. In this model the image flow is given by $u = a_1$ and $v = a_2$ where a_1 and a_2 are functions of V_X and V_Y, respectively.

3. Locally-rigid tracking approach

The rigid translation model allows the constraint of the scene flow in the image domain. Hence, the scene flow problem can be formulated as finding the 3D motion field that best explains the observed intensity data, exploiting the 2D parameterization. Assuming a scene composed of independently but locally-rigid 3D parts, we state below the scene flow computation inspired by the Lucas-Kanade [33] formulation for local optical flow.

3.1. Lucas-Kanade framework

The goal of Lucas-Kanade algorithm is to align a template image $T(\mathbf{x})$ to an input image $I(\mathbf{x})$, so that, it allows to compute optical flow or to track an image patch between two frames. Following [13], this problem can be stated as finding the parameter vector $\mathbf{P} = (p_1, ..., p_n)^T$ which minimizes a squared sum of intensities between the template T an the current image I:

$$\mathbf{P} = \arg\min_{\mathbf{P}} \sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x}; \mathbf{P})) - T(\mathbf{x})]^2, \tag{13}$$

where the warp function $\mathbf{W}(\mathbf{x}; \mathbf{P})$ maps each template pixel $\mathbf{x}$ to a pixel on the image. The warp function can be defined using any motion model, *e.g.*, one of those presented in Section 2.

3.2. Brightness consistency and locally-rigid warp

Under brightness constancy assumption, points $\mathbf{X}$ at time $t-1$ and $\mathbf{X}' = \mathbf{X} + \mathbf{V}$ at time t are projected with the same intensity on the image according to

$$I^{t-1}(\hat{\mathbf{M}}(\mathbf{X})) = I^t(\hat{\mathbf{M}}(\mathbf{X}')), \tag{14}$$

where $\hat{\mathbf{M}}$ is the projective function that maps points in the space to the image plane. Hence, if $I^t(\mathbf{x})$ is the intensity of the image pixel $\mathbf{x}$ at time t , the brightness consistency can be written in the image domain as

$$I^{t-1}(\mathbf{x}) = I^t(\mathbf{x}'), \tag{15}$$

with $\mathbf{x}$ and $\mathbf{x}'$ corresponding to the projections of 3D points $\mathbf{X}$ and $\mathbf{X}'$, respectively. Let T stand for the previous frame I^{t-1} (intensity template) and I stand for the current frame I^t, then for a set of visible points $\mathbf{S}$ of the rigid surface, under translation $\mathbf{V}$ and satisfying the brightness constancy assumptions, we have:

$$\sum_{\mathbf{X} \in \mathbf{S}} \left[I\left(\hat{\mathbf{M}}(\mathbf{X} + \mathbf{V})\right) - T\left(\hat{\mathbf{M}}(\mathbf{X})\right) \right]^2 = 0. \tag{16}$$

If the set $\mathbf{s}$ is the projection of $\mathbf{S}$ on the image and $\mathbf{W}(\mathbf{x}; \mathbf{V})$ is a warp function mapping each pixel $\mathbf{x}$ off the image T to the corresponding pixel $\mathbf{x}'$ on the image I, (16) becomes

$$\sum_{\mathbf{x} \in \mathbf{s}} [I(\mathbf{W}(\mathbf{x}; \mathbf{V})) - T(\mathbf{x})]^2 = 0. \tag{17}$$

The warp function that satisfies (17) can be defined as

$$\mathbf{W}(\mathbf{x}; \mathbf{V}) = \mathbf{x} + \Delta(\mathbf{x}; \mathbf{V}) = \mathbf{x}' \tag{18}$$

with the delta function modeling the image flow induced by translation $\mathbf{V}$. Following (8) the image flow induced over each surface pixel is given by:

$$\Delta(\mathbf{x}; \mathbf{V}) = \frac{1}{Z(\mathbf{x})} \begin{pmatrix} f_x & 0 & c_x - x \\ 0 & f_y & c_y - y \end{pmatrix} \begin{pmatrix} V_X \\ V_Y \\ V_Z \end{pmatrix}, \tag{19}$$

where $Z(\mathbf{x})$ is the depth of pixel $\mathbf{x}$ (Z component of $\mathbf{X}$).

3.3. Tracking in the intensity image

We assume a locally-rigid scene where 2D motions on a image region are generated by the translational motion of a rigid surface. In this case, the warp function given by equation (18)

allows to estimate the 3D motion in a Lucas-Kanade formulation, where the parameter vector $\mathbf{P}$ corresponds to the 3D motion vector $\mathbf{V}$ that describes the surface translation.

To get a reliable estimation of the 3D motion vector the image region must be sufficiently large to constrain the solution, yet the larger the region of summation the more likely it is to contain multiple motions. Besides, the brightness constancy assumption could be violated due to factors as noise sensor, illumination changes, specular surfaces, etc. To reduce the influence of outliers we use the function $\psi(x^2) = \sqrt{x^2 + \epsilon^2}$ [34], a differentiable variant of the L_1-norm. Therefore, we formulate the tracking problem as finding the velocity vector $\mathbf{V}$ that minimizes

$$\sum_{\mathbf{x}} \psi\left(E_I(\mathbf{x};\mathbf{V})^2\right), \tag{20}$$

where $E_I(\mathbf{x};\mathbf{V}) = I(\mathbf{W}(\mathbf{x};\mathbf{V})) - T(\mathbf{x})$ stands for the *intensity difference image*. In our approach, we assume that all template pixels belong to the same rigid surface in the scene, which performs a translational motion between consecutive frames.

Minimization of expression (20) is a nonlinear optimization problem, which can be solved by IRLS [35]. Considering that an initial estimate of $\mathbf{V}$ is known and using the first order Taylor expansion on the intensity function I, equation (20) becomes the problem of finding the incremental $\Delta\mathbf{V}$ that minimizes

$$\sum_{\mathbf{x}} \psi\left(\left(E_I(\mathbf{x};\mathbf{V}) + \nabla_I \frac{\partial \mathbf{W}}{\partial \mathbf{V}} \Delta\mathbf{V}\right)^2\right), \tag{21}$$

where $\nabla_I = (\partial I/\partial x, \partial I/\partial y)$ is the row vector gradient evaluated at $\mathbf{W}(\mathbf{x};\mathbf{V})$ and the term $\partial\mathbf{W}/\partial\mathbf{V}$ is the Jacobian $\mathbf{J}$ of the warp. If $\mathbf{W}(\mathbf{x};\mathbf{V}) = (W_x, W_y)^T$ the Jacobian is given by:

$$\mathbf{J} = \begin{pmatrix} \frac{\partial W_x}{\partial V_X} & \frac{\partial W_x}{\partial V_Y} & \frac{\partial W_x}{\partial V_Z} \\ \frac{\partial W_y}{\partial V_X} & \frac{\partial W_y}{\partial V_Y} & \frac{\partial W_y}{\partial V_Z} \end{pmatrix} = \frac{1}{Z(\mathbf{x})} \begin{pmatrix} f_x & 0 & c_x - x \\ 0 & f_y & c_y - y \end{pmatrix}. \tag{22}$$

Finding the minimum of expression (21) requires an iterative solution. Taking the partial derivative with respect to $\Delta\mathbf{V}$ gives:

$$2\sum_{\mathbf{x}} \Psi(\mathbf{x};\mathbf{V};\Delta\mathbf{V})(\nabla_I\mathbf{J})^T (E_I(\mathbf{x};\mathbf{V}) + (\nabla_I\mathbf{J})\Delta\mathbf{V}), \tag{23}$$

where for brevity $\Psi(\mathbf{x};\mathbf{V};\Delta\mathbf{V}) = \psi'(E_I(\mathbf{x};\mathbf{V}) + (\nabla_I\mathbf{J})\Delta\mathbf{V})$ with $\psi'(x)$ the derivative of the robust norm. At the minimum of (21), equation (23) is zero, so that, the incremental scene flow $\Delta\mathbf{V}$ can be computed by the following iterative procedure:

1. Initialize: $\Delta\mathbf{V} = 0$.
2. Compute $\Psi(\mathbf{x};\mathbf{V};\Delta\mathbf{V})$ and then the velocity update

$$\Delta\mathbf{V} = -H^{-1} \sum_{\mathbf{x}} \Psi(\mathbf{x};\mathbf{V};\Delta\mathbf{V})(\nabla_I\mathbf{J})^T E_I(\mathbf{x};\mathbf{V}), \tag{24}$$

where H, the Gauss-Newton approximation of the Hessian, is given by:

$$H = \sum_{\mathbf{x}} \Psi(\mathbf{x};\mathbf{V};\Delta\mathbf{V})(\nabla_I\mathbf{J})^T (\nabla_I\mathbf{J}), \tag{25}$$

3. Update step: $\mathbf{V} \leftarrow \mathbf{V} + \Delta\mathbf{V}$.
4. If $\Delta\mathbf{V} < \epsilon$ stop, otherwise goto 1.

Over each iteration the Hessian matrix can be expressed as

$$H = \sum_{\mathbf{x}} \frac{\psi_I'(\mathbf{x};\mathbf{V};\Delta\mathbf{V})}{Z(\mathbf{x})^2} \begin{pmatrix} I_x^2 & I_xI_y & I_xI_\Sigma \\ I_xI_y & I_y^2 & I_yI_\Sigma \\ I_xI_\Sigma & I_yI_\Sigma & I_\Sigma^2 \end{pmatrix}, \tag{26}$$

with $I_\Sigma = -(xI_x + yI_y)$. A reliable solution of (20) requires that matrix H given by (26) must be above of the noise of the intensity images and well conditioned. To construct H each 3×3 gradient based matrix is weighted by the square inverse depth of the pixel. Since the depth of each template pixel is required to computed the solution only template pixels with valid depth measures should be considered.

4. Tracking in intensity and depth

In the previous Section we have solved for local scene flow by using a tracking approach in a Lucas-Kanade framework. Thus, assuming a local rigidity of the scene, the parameter vector of the warp function corresponds to the 3D motion vector. The depth value of each image pixel is only used to compute the image flow induced by the estimated scene flow. However, in this approach the motion vector is constrained using only brightness data and the information from the depth sensor is not fully exploited.

Taking advantage of the structure information captured by the depth sensor we propose to incorporate a constraint based on the observed depth velocity. The new constraint is computed from the squared difference between the estimated depth of each template pixel, given by V_Z, and the current depth measure provided by the sensor.

4.1. Depth velocity constraint

Since the scene is assumed to be composed by locally-rigid 3D parts performing translation, the motion of a set of 3D points belonging to a rigid surface can be expressed as $\mathbf{X}' = \mathbf{X} + \mathbf{V}$. Therefore, the depth of each surface point between consecutive frames, is related by the Z component of the local scene flow as $Z' = Z + V_Z$. If $Z^t(\mathbf{x})$ is the depth of the image pixel $\mathbf{x}$ at time t, the depth data must satisfy:

$$Z^{t-1}(\hat{\mathbf{M}}(\mathbf{X})) + V_Z = Z^t(\hat{\mathbf{M}}(\mathbf{X}')), \tag{27}$$

where $\hat{\mathbf{M}}$ is the projective function and, $\mathbf{x}$ and $\mathbf{x}'$ correspond to the projections of 3D points $\mathbf{X}$ and $\mathbf{X}'$, respectively. Let T_Z stand for the previous depth frame Z^{t-1} (depth template) and Z stand for the current frame Z^t. Then for a set of visible points $\mathbf{S}$ of a rigid surface we have:

$$\sum_{\mathbf{X}\in\mathbf{S}} \left[Z\left(\hat{\mathbf{M}}(\mathbf{X}+\mathbf{V})\right) - \left(T_Z\left(\hat{\mathbf{M}}(\mathbf{X})\right) + V_Z\right)\right]^2 = 0. \tag{28}$$

Using the warp function (18) equation (28) can be expressed in the image domain as follows:

$$\sum_{\mathbf{x}\in\mathbf{s}} \left[Z(\mathbf{W}(\mathbf{x};\mathbf{V})) - (T_Z(\mathbf{x}) + V_Z)\right]^2 = 0, \tag{29}$$

where $\mathbf{s}$ is the projection of set $\mathbf{S}$ on the image.

4.2. Tracking algorithm in intensity and depth

Using brightness and depth velocity constraints the scene flow computation can be formulated as finding de 3D motion vector that best explains the observed intensity and depth data over an image region. Therefore, we formulate the local scene flow as the optimization problem given by:

$$\mathbf{V} = \arg\min_{\mathbf{V}} \sum_{\mathbf{x}} E_I(\mathbf{x};\mathbf{V})^2 + \lambda E_Z(\mathbf{x};\mathbf{V})^2, \tag{30}$$

where

$$E_I(\mathbf{x};\mathbf{V}) = I(\mathbf{W}(\mathbf{x};\mathbf{V})) - T(\mathbf{x}), \tag{31}$$

$$E_Z(\mathbf{x};\mathbf{V}) = Z(\mathbf{W}(\mathbf{x};\mathbf{V})) - \left(T_Z(\mathbf{x}) + D^T\mathbf{V}\right), \tag{32}$$

and $\lambda = \sigma_I^2/\sigma_Z^2$, with σ_I^2 and σ_Z^2 the noise variance in intensity and depth sensors, respectively. The row vector $D^T = (0,0,1)$ isolates the Z component of the scene flow.

The first term $E_I(\mathbf{x};\mathbf{V})^2$ is the *intensity term* which measures how consistent is the image flow (induced by $\mathbf{V}$) of each template pixel with the input intensity images. The second term $E_Z(\mathbf{x};\mathbf{V})^2$ is the *depth term* which measures the consistency of the estimated scene flow $\mathbf{V}$ with the input depth data. A reliable solution of the optimization problem (30) ensures a scene flow consistent with both intensity and depth data.

Assuming an initial estimation of $\mathbf{V}$, each optimization step searches for $\Delta\mathbf{V}$ which minimizes

$$\sum_{\mathbf{x}} E_I(\mathbf{x};\mathbf{V}+\Delta\mathbf{V})^2 + \lambda E_Z(\mathbf{x};\mathbf{V}+\Delta\mathbf{V})^2. \tag{33}$$

In order to solve for the incremental $\Delta\mathbf{V}$, (33) is linearized using the first order Taylor expansions on I and Z

$$\sum_{\mathbf{x}} [E_I(\mathbf{x};\mathbf{V}) + (\nabla_I\mathbf{J})\Delta\mathbf{V}]^2 + \lambda\left[E_Z(\mathbf{x};\mathbf{V}) + \left(\nabla_Z\mathbf{J} - D^T\right)\Delta\mathbf{V}\right]^2 \tag{34}$$

where $\nabla_I = (\partial I/\partial x, \partial I/\partial y)$ and $\nabla_Z = (\partial Z/\partial x, \partial Z/\partial y)$, both evaluated at $\mathbf{W}(\mathbf{x};\mathbf{V})$, and with $\mathbf{J}$ the Jacobian of the warp function given by (22). Now, taking the partial derivative of (34) with respect $\Delta\mathbf{V}$, setting this expression to zero and solving for $\Delta\mathbf{V}$, the update rule can be stated as

$$\Delta\mathbf{V} = -H^{-1}\sum_{\mathbf{x}} (\nabla_I\mathbf{J})^T E_I(\mathbf{x};\mathbf{V}) + \lambda\left(\nabla_Z\mathbf{J} - D^T\right)^T E_Z(\mathbf{x};\mathbf{V}) \tag{35}$$

where matrix H is given by

$$H = \sum_{\mathbf{x}} (\nabla_I\mathbf{J})^T(\nabla_I\mathbf{J}) + \lambda\left(\nabla_Z\mathbf{J} - D^T\right)^T\left(\nabla_Z\mathbf{J} - D^T\right). \tag{36}$$

The Hessian matrix can be expressed as

$$H = \sum_{\mathbf{x}} \frac{1}{Z(\mathbf{x})^2}\begin{pmatrix} I_x^2 + \lambda Z_x^2 & I_xI_y + \lambda Z_xZ_y & I_xI_\Sigma + \lambda Z_x(Z_\Sigma - 1) \\ I_xI_y + \lambda Z_xZ_y & I_y^2 + \lambda Z_y^2 & I_yI_\Sigma + \lambda Z_y(Z_\Sigma - 1) \\ I_xI_\Sigma + \lambda Z_x(Z_\Sigma - 1) & I_yI_\Sigma + \lambda Z_y(Z_\Sigma - 1) & I_\Sigma^2 + \lambda(Z_\Sigma - 1)^2 \end{pmatrix}, \tag{37}$$

with $I_\Sigma = -(xI_x + yI_y)$ and $Z_\Sigma = -(xZ_x + yZ_y)$. A reliable solution of (30) requires that matrix H given by (37) must be both above of the noise of the images (intensity and depth) and well conditioned. Unlike Lucas-Kanade approach with a pure translation motion model, where the aperture problem can be analyzed using the 2×2 gradient covariance matrix of the image [36], equation (37) state a new formulation. Now intensity and depth data are combined to determine the local scene flow and the image motion.

4.3. Implementation details

In order to compute the local scene flow at a given pixel $\mathbf{x_0}$, the proposed approach assumes an aligned pair of intensity and depth sequences, $I(\mathbf{x},t)$ and $Z(\mathbf{x},t)$, respectively. Taking the former frames, the intensity template $T(\mathbf{x})$ and the depth template $T_Z(\mathbf{x})$ are defined by selecting a $(2\omega_x+1)\times(2\omega_y+1)$ neighborhood $\Omega(\mathbf{x}_0)$, centering at $\mathbf{x_0}$. The initial estimate of $\mathbf{V}$ is set to zero and thereby in the first iteration the warp function obeys $\mathbf{W}(\mathbf{x};\mathbf{V}) = \mathbf{x}$ for all $\mathbf{x} \in \Omega(\mathbf{x}_0)$. In formulation (30), we have not used a robust norm in order to get a closed solution for $\Delta\mathbf{V}$. However in our implementation we apply the same variant of the L_1-norm that in (20) to each one of the terms:

$$\mathbf{V} = \arg\min_{\mathbf{V}} \sum_{\mathbf{x}\in\Omega(\mathbf{x}_0)} \psi\left([I(\mathbf{W}(\mathbf{x};\mathbf{V})) - T(\mathbf{x})]^2\right) + \lambda\,\psi\left(\left[Z(\mathbf{W}(\mathbf{x};\mathbf{V})) - \left(T_Z(\mathbf{x}) + D^T\mathbf{V}\right)\right]^2\right) \tag{38}$$

with $\epsilon = 0.001$. Setting parameter λ to be zero, the intensity and depth based objective function (38) becomes the only intensity tracking formulation of Section 3.3. Since it is possible that the depth sensor does not provide a measure for each pixel on $\Omega(\mathbf{x}_0)$, only pixels with a valid depth measure are taken into account for both $Z(\mathbf{W}(\mathbf{x};\mathbf{V}))$ and $T_Z(\mathbf{x})$.

Coarse-to-fine estimation. The proposed algorithm solves for the scene flow estimation problem as local nonlinear optimization by assuming an initial estimate of $\mathbf{V}$ at pixel $\mathbf{x}$. In order to deal with long-range motion, a coarse-to-fine strategy is required. We perform a multi-resolution technique to estimate scene flow under large displacement, where the local 3D motion estimated at a coarser level is used to locally warp the input intensity and depth images toward the intensity and depth templates at the next finer level.

We construct an image pyramid with a downsampling factor of 2. For the intensity images we build a Gaussian pyramid where the standard deviation of the Gaussian anti-aliasing filter is set to be 1/2. In order to not propagate invalid depth measures, the depth pyramid is constructed by averaging pixels in non-overlapped neighborhoods of 2×2 where only pixels with valid depth are used. For both intensity and depth pyramids, each level is recursively downsampled from its nearest lower level. Using a pyramid with levels $L = 0, 1, ..., M$, where $L = 0$ is the original resolution, the local scene flow estimation starts from the highest level M with the initial condition $\mathbf{V} = 0$. In each pyramid level L the computation is performed over the image neighborhood given by $\mathbf{\Omega}(\mathbf{x}/2^L)$ and the warp function (18) is modified by scaling the focal length and the camera center by

the factor 2^L. The computed scene flow is directly propagated to the next lower level.

5. Experiments

5.1. Middlebury datasets

In order to evaluate the performance of our method we have conducted experiments using an existing scene flow benchmark. The proposed algorithm that allows the constraint of the scene flow in intensity and depth is named SF_{ID} and its version using only intensity constraints ($\lambda = 0$) is SF_I. We compare our method with three versions of the Lukas-Kanade tracker (KLT) as well as with other scene flow algorithms.

As in [9, 18, 19] Middlebury stereo datasets [37], *Cones* and *Teddy*, were used for the experiments. Using images of one of these datasets is equivalent to a fixed camera observing a moving object along X axis. We took image 2 of each dataset as the first frame and image 6 as the second frame, both in quarter-size. Parameters of the camera setup were estimated and the depth data was generated from the disparity map of each image.

The ground truth for the scene flow is constant and given by the baseline of the stereo setup. On the other hand, the ground truth for the optical flow is given by the disparity map (used as depth image) from frame 1, as show in Figure 1. In this dataset all movements, both in the 3D scene and on the image, are purely horizontal.

5.1.1. Error measures

In order to measure errors in 2D we compute the *root mean squared error* of the optical flow (RMS_{OF}) and the *average angle error* (AAE_{OF}). Besides, as in [37], we use the statistic RX of RMS_{OF} to observe the percentage of pixels that have an error measure above X pixels, for the values $X = 1.0$ and $X = 5.0$. Since errors on the image do not necessarily correspond with errors in the scene [19], we evaluate both 2D and 3D errors in the experiments. In order to asses errors in the scene flow we compute the *normalized root mean squared error* of the 3D flow vector ($NRMS_V$). The $NRMS_V$ definition is close to that of [20], but we normalize the rms error by the (constant) magnitude of the scene flow ground truth which is the same for both datasets. This normalization allows calculating the error as percentage of the ground truth. As in the 2D case, we compute the statistic RX of $NRMS_V$, now for the percentages $X = 5\%$ and $X = 20\%$.

Unlike optical flow, there is no standard metric to assess scene flow algorithms. Each author uses an error metric with or without normalization. In order to compare our method with other scene flows approaches we choose the *normalized root mean squared error* of the optical flow ($NRMS_{OF}$) because it is available or can be derivate from the selected previous works. We normalize our results by the optical flow range of each dataset, given by 42.5 for *Teddy* and 48.5 for Cones.

5.1.2. Comparison with sparse optical flow methods

Lucas-Kanade algorithm is the most frequently used method to compute sparse optical flow and to get 2D trajectories. Even different motions models can be used in this framework (e.g., affine, homography) the pure translation model [36] is the most successfully used. Therefore, we select for comparison the C pyramidal implementation of KLT by Bouguet [38] and our own pyramidal implementation with a robust norm (KLT_R). We also implemented a pyramidal version of the orthographic intensity and depth tracker [26] provided of a robust norm, which we call (KLT_O). All algorithms were configured to use a fixed window size of 11×11, in a Gaussian pyramidal decomposition of 5 levels. In the experiments we performed the computation over a regular grid covering the 85% of the image area.

		SF_{ID}	SF_I	KLT	KLT_R	KLT_O
Teddy	RMS_{OF}	2.71	2.92	7.21	4.18	4.62
	R1.0	9.54	12.9	40.2	23.7	28.9
	R5.0	2.50	3.70	21.9	13.25	19.1
	AAE_{OF}	1.17	1.38	1.67	1.18	1.48
Cones	RMS_{OF}	2.32	2.31	4.70	5.20	4.02
	R1.0	16.3	16.7	39.9	33.1	40.3
	R5.0	2.15	4.29	17.6	17.2	14.6
	AAE_{OF}	1.14	1.28	1.24	1.84	1.42

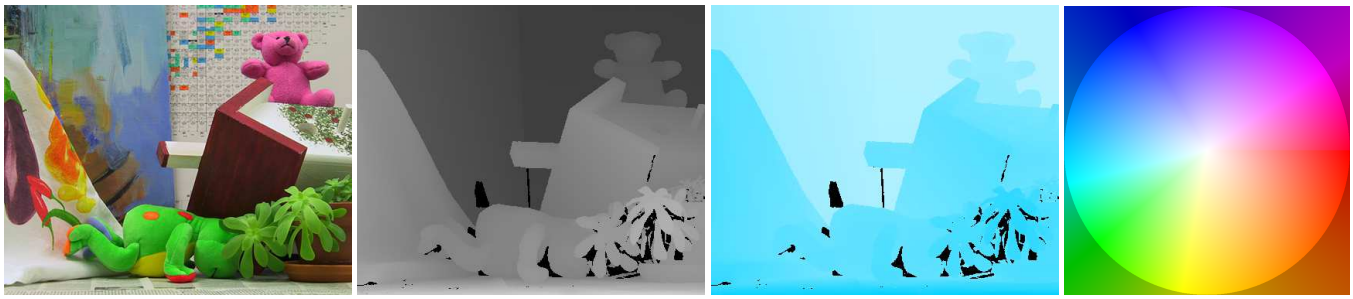
Table 1: Results in the image domain: optical flow errors.

		SF_{ID}	SF_I	KLT	KLT_R	KLT_O
Teddy	$NRMS_V$	11.4	60.6	74.1	41.2	41.2
	R5%	18.6	29.1	40.2	37.2	38.3
	R20%	7.06	12.6	22.2	20.4	23.2
Cones	$NRMS_V$	10.8	51.2	89.9	95.6	93.7
	R5%	15.6	28.5	35.9	38.6	39.8
	R20%	2.89	6.77	16.0	14.5	14.9

Table 2: Results in 3D: scene flow errors.

Results over non-occluded regions are presented in Table 1 for the optical flow. The proposed algorithm SF_{ID} and its variant SF_I outperform other trackers in the estimation of the optical flow magnitude. Errors in orientation are very close. It is important to note that using the depth information as proposed by [26] does not improve the optical flow estimation, indeed the KLT_R without including depth information performs better. Statistic R5.0 shows that the number of outliers (above 5 pixels) is reduced when the proposed pixel motion model is used even under the large displacements of the dataset.

The most direct way to compute scene flow when a depth camera and an optical flow algorithm are available is by inferring the 3D motion using the resulting image motion and the provided depth data. We follow this procedure to estimate the scene flow for each optical flow method. Results for the scene flow are presented in Table 2. The SF_{ID} algorithm achieves the most accurate results. Statistics R5% and R20% allows to do a more detailed comparison between the methods since $NRMS_V$ is highly sensitive to outliers. As can be seen from Tables 1 and 2, even small pixel deviations in the image domain can have a strong influence in the 3D error computed in the scene. Comparing the results of SF_{ID} and KLT_O, it is clear that the pro-



(a) (b) (c) (d)

Figure 1: Teddy stereo dataset [37]. (a) Color image, (b) depth image (disparity image in the dataset), (c) optical flow ground truth and (d) flow color coding.

	SF_{ID}			KLT_R		
	Tex	*Utex*	*DD*	*Tex*	*Utex*	*DD*
RMS_{OF}	4.99	3.11	6.91	5.75	7.20	7.05
R1.0	16.5	39.8	32.5	38.9	58.8	68.7
$NRMS_V$	23.2	10.9	26.5	96.7	202	188
R5%	12.1	28.5	25.1	32.0	51.5	69.6

Table 3: Errors by regions.

		SF_{ID}	[18]	[19]	[9]
Teddy	RMS_{OF}	0.067	0.062	0.028	0.090
	AAE_{OF}	2.73	0.51	1.01	5.04
Cones	RMS_{OF}	0.046	0.057	0.030	0.11
	AAE_{OF}	2.22	0.69	0.39	5.02

Table 4: Comparison of scene flow algorithms in 2D.

posed warp function allows to estimate a more accurate scene flow than the pure translation model.

In order to observe the performance in specific regions, the scene flow is computed on textured and untextured regions, as well as on depth discontinuities. Points for texture analysis are chosen by observing the minimum eigenvalue (λ_{min}) of the Hessian matrix of I on a region. Higher values of λ_{min} are used to select textured (Tex) regions while lower values of λ_{min} define untextured (Utex) regions. Regions with depth discontinuities (DD) are selected by finding higher values in the depth image gradient. For each of these regions the averaged error on *Teddy* and *Cones* images is presented in Table 3. For all regions our method outperforms the robust norm KLT version and results show that better regions to track correspond to textured regions avoiding strong depth discontinuities. The accuracy is reduced on DD regions since the number of outliers increases due to occluded regions and unmeasured depth points. Although scene flow error R5% is strongly affected on DD, the use of depth data reduce the number of outliers in the image domain w.r.t. OF_R. In average, errors on untextured regions are lower but the percentage of points with inaccurate optical flow or scene flow is the highest, as can be seen in Table 3 for R1.0 and R5%. Finally, on textured regions the $NRMS_V$ error is increased due to the presence of depth discontinuities, however, observing R1.0 and R5% it is evident that precision on this regions is higher, both in optical flow and scene flow.

5.1.3. Comparison with other scene flow methods

There is not a standard procedure or error measure to compare scene flow methods. Therefore, in order to compare the proposed SF_{ID} algorithm we compute the $NRMS_{OF}$ and AAE_{OF} in *Teddy* and *Cones* datasets, as in [18], [19] and [9]. To allow the comparison with dense methods we compute the scene flow for all pixels. Results are presented in Table 4.

The proposed method SF_{ID} outperforms [9] which is based on particle filtering using intensity and depth. The particle filter is benefited from the constant scene flow of the datasets, since most likely hypothesis are diffused over the whole image. On the other hand, comparison with stereo-based techniques is not straightforward since they use closer images of the Middlebury datasets and the depth information (disparity) must be estimated. In [18] and [19], images 2 and 6 are taken as the stereo pair at time $t-1$ and images 4 and 8 correspond to the stereo pair at time t. Accordingly, the optical flow ground truth is half of the disparity and there are fewer occluded regions. Moreover, since [19] uses a total variation regularization in 3D this algorithm is also favored by the constant scene flow of the dataset. Results of the SF_I algorithm are good enough even it does not use a global regularization as in [18] and [19], or a large set of hypotheses as in [9]. In order to get more accurate estimation the scene flow computation must be concentrated in good regions to track.

A non-optimized version of the proposed local scene flow algorithm processed the Middlebury dataset, on a single core desktop machine in under 5 s, as opposed to 5 h for [18] or 10 min for [9] (run time was not reported in [19]).

5.2. Kinect images

Some experiments were performed with image sequences from a Microsoft Kinect sensor where intensity and depth images were accessed using the libfreenect library in C++. We used the Kinect calibration toolbox [40] to get intrinsic and extrinsic parameters of the depth and RGB camera pair and to perform the image alignment. Aligned pairs of intensity and

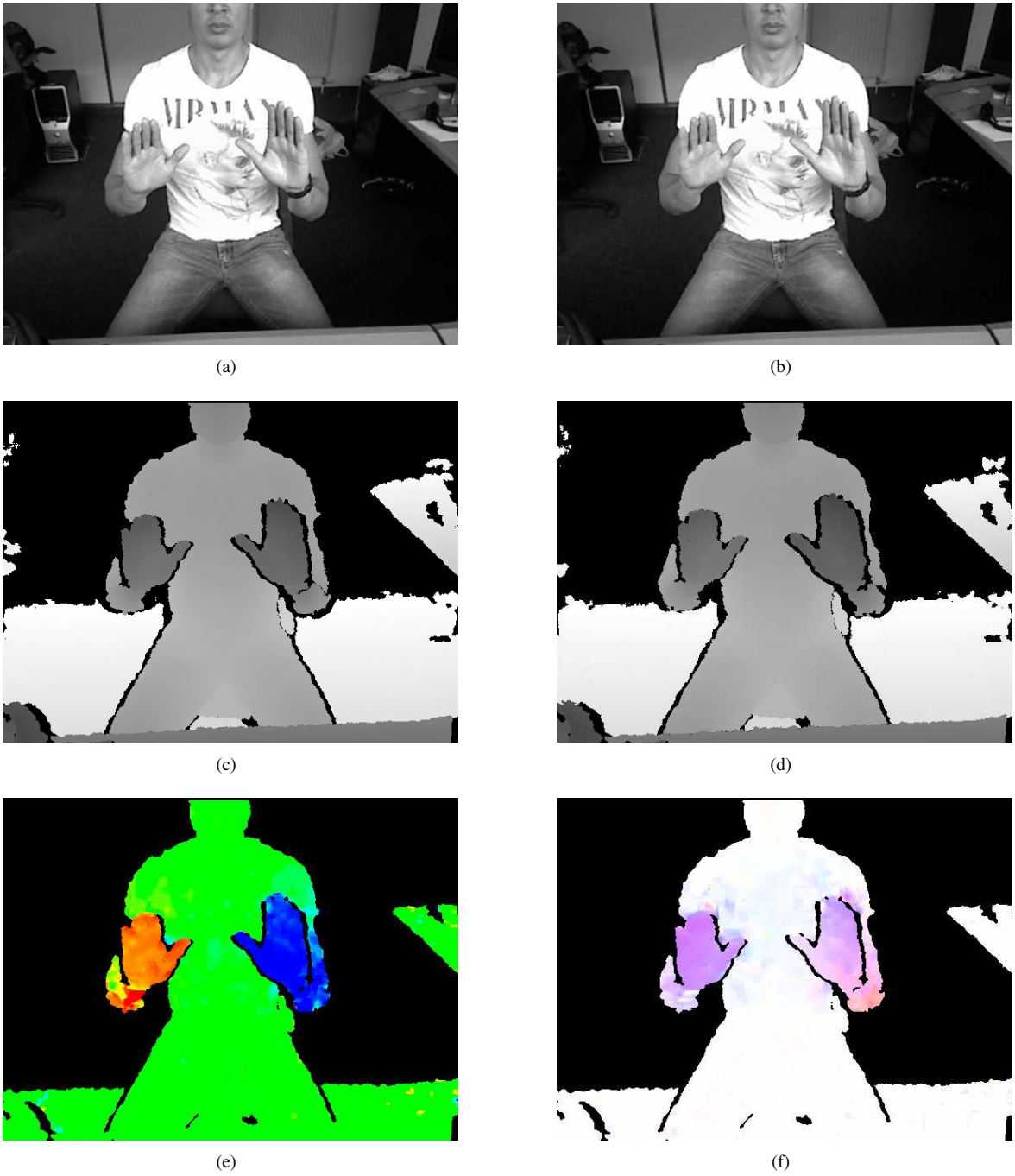


Figure 2: Results from a pair of Kinect images. (a-b) frames at times $t-1$ and t, (c-d) depth images at times $t-1$ and t, (e) Z component of the scene flow and (f) scene flow projection on the image (optical flow). The scene flow is computed for each pixel within 2 meters from the sensor, results for farther pixels are presented in black as well as the motion estimation in regions where most pixels have no valid depth. Depth changes (Z component of the scene flow) are encoded in a cold-to-warm color map for the range [-2.5,2.5] cm, where the zero velocity is green, blue and magenta colors represent negative velocities (approaching pixels) and red and yellow colors are positives velocities. Optical flow is presented using the color coding shown in Figure 1(d) [39].

depth images are shown in Figure 2. Depth data is encoded using gray levels, where lighter pixels correspond to closer points from the sensor. Unmeasured regions are encoded in black and they correspond to occluded or out of range regions to the depth camera, or to points belonging to reflective surfaces.

In the first experiment we computed a local scene flow for each pixel having a valid depth measure and being within 2 meters from the camera. For the computation we used a window size of 11×11 in a pyramidal decomposition of 2 levels. Figure 2 shows the results of the motion field estimation between a pair of images where a person is performing a motion with both hands. In the sequence, the left (on the image) hand is moving

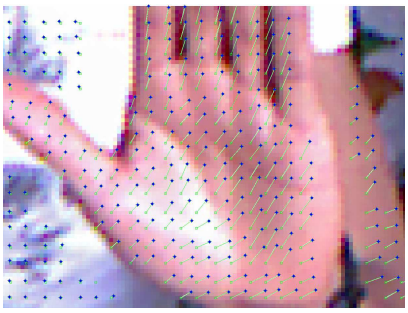


Figure 3: Scene flow projection on the image.

away from the sensor while the right hand is approaching. The resulting Z component of the scene flow and optical flow and are presented in Figures 2(e) and 2(f), respectively.

Points on the hand surface present approximately the same scene flow while the optical flow magnitude is function of the depth. Using only the optical flow is not possible to distinguish the different types of movement that each hand performs. This ambiguity can be resolved by means of the scene flow, e.g. using its Z component as illustrated in Figure 2(e). Since the scene flow is computed locally a reliable solution can not be found over untextured regions, e.g., the t-shirt, or in depth discontinuities where most of depth data is missing. However, the solution on slightly textured regions, as the hands, is good enough. Figure 3 shows an sparse projection of the scene flow on an image region containing a hand, where missing values correspond to regions without enough depth information.

In the second experiment we track a set of 4 image points selected manually, over a sequence of 200 frames. Figure 4(a) shows the chosen points which are located in both arms on slightly textured regions. Considering a neighborhood around each point, a local scene flow is computed between consecutive frames to determine the new location of the interest point. Although the skin color is not textured enough, the tracking can be performed with help from the depth data. Some images of the tracking experiment are presented in Figure 4(a)-(f).

6. Conclusions

We have proposed a novel approach to compute local scene flow by tracking surface patches in intensity and depth. Using a rigid translation flow model we state the scene flow computation as a 2D tracking problem, achieving the computation of the 3D motion vector that best explains the provided intensity and depth data. Our solution is formulated in a Lucas-Kanade framework provided of a L_1-norm approximation for each data term, allowing to deal with outliers and to perform a coarse-to-fine solution. Although we have chosen a specific flow model on the image, using some assumptions, the presented formulation for the scene flow problem is valid with other motion models wherever motion field $\mathbf{V} = (V_X, V_Y, V_Z)$ can be computed from the vector parameter $\mathbf{P}$. Throughout some experiments we demonstrated the validity of our method and we showed that it outperforms other Lucas-Kanade tracker for optical flow and scene flow computation. We also perform comparison with other scene flow techniques with very encouraging results. As we simultaneously solve for the 2D tracking and for the scene flow this method could be used directly in motion analysis tasks to generate accurate 3D trajectories or define scene flow based descriptors. Unlike previous scene flow methods, the proposed approach is "near" real time with 5s processing time per frame.

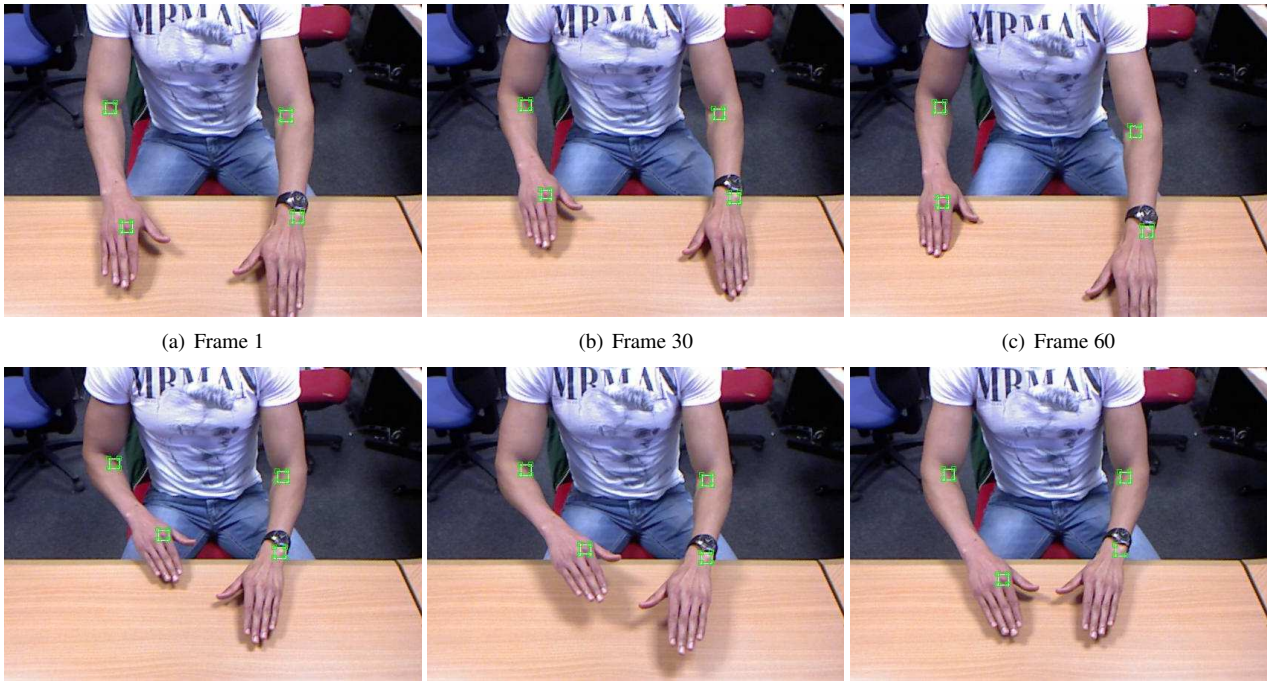
We are currently exploring a variational formulation of the scene flow computation using intensity and depth in order to improve estimation on untextured regions and those close to strong depth discontinuities. Besides we are investigating how to define scene flow based descriptors to perform action or gestures recognition. Since our method require the inversion of a 3×3 matrix which is function of the gradients of intensity and depth images, we expect to obtain a criterion for selecting good regions to compute its local scene flow.

Acknowledgement

Part of this work was previously presented [41] in the 2nd International Workshop on Human Activity Understanding from 3D Data (HAU3D12), which was held on in conjunction with the IEEE Conference on Computer Vision and Pattern Recognition, Providence, 2012.

References

[1] S. Vedula, S. Baker, P. Rander, R. Collins, Three-dimensional scene flow, in: International Conference on Computer Vision, 1999.
[2] I. Laptev, T. Lindeberg, Space-time interest points, in: European Conference on Computer Vision, 2003.
[3] J. Niebles, C. Chen, L. Fei-Fei, Modelling temporal structure of decomposable motion segments for activity classification, in: European Conference on Computer Vision, 2010.
[4] M. Raptis, S. Soatto, Tracklet descriptors for action modelling and video analysis, in: European Conference on Computer Vision, 2010.
[5] R. Messing, C. Pal, H. Kautz, Activity recognition using the velocity histories of tracked keypoints, in: International Conference on Computer Vision, 2009, 2009.
[6] P. Matikainen, M. Hebert, R. Sukthankar, Trajectons: Action recognition through the motion analysis of tracked features, in: International Conference on Computer Vision Workshops, 2009.
[7] H. Wang, A. Kläser, C. Schmid, C.-L. Liu, Action recognition by dense trajectories, in: Conference on Computer Vision & Pattern Recognition, 2011, 2011.
[8] A. Letouzey, B. Petit, E. Boyer, Scene flow from depth and color images, in: British Machine Vision Conference, 2011, 2011.
[9] S. Hadfield, R. Bowden, Kinecting the dots: Particle based scene flow form depth sensors, in: International Conference on Computer Vision, 2011.
[10] W. Li, Z. Zhang, Z. Liu, Action recognition based on a bag of 3D points, in: Conference on Computer Vision and Pattern Recognition Workshops, 2010.
[11] B. Ni, G. Wang, P. Moulin, RGBD-HuDaAct: A color-depth video database for human daily activity, in: Conference on Computer Vision and Pattern Recognition Workshops, 2011.
[12] A. Bobick, J. Davis, The representation and recognition of action using temporal templates, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (2001) 257–267.
[13] S. Baker, I. Matthews, Lucas-Kanade 20 years on: A unifying framework, International Journal of Computer Vision 56 (2004) 221–255.
[14] S. Vedula, S. Baker, P. Render, R. Collins, T. Kanade, Three-dimensional scene flow, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (2005) 275–280.



(a) Frame 1 (b) Frame 30 (c) Frame 60

(d) Frame 90 (e) Frame 120 (f) Frame 150

Figure 4: Tracking a set of points by computing local scene flow.

[15] Y. Zhang, C. Kambhamettu, On 3D scene flow and structure estimation, in: Conference on Computer Vision and Pattern Recognition, 2001.

[16] R. Li, S. Sclaroff, Multi-scale 3D scene flow from binocular stereo, Computer Vision and Image Understanding 110 (2008) 75–90.

[17] A. Wedel, C. Rabe, T. Vaudrey, T. Brox, U. Franke, D. Cremers, Efficient dense scene flow from sparse of dense stereo data, in: European Conference on Computer Vision, 2008.

[18] F. Huguet, F. Devernay, A variational method for scene flow estimation from stereo sequences, in: International Conference on Computer Vision, 2007.

[19] T. Basha, Y. Moses, N. Kiryati, Multi-view scene flow estimation: A view centered variational approach, in: Conference on Computer Vision and Pattern Recognition, 2010.

[20] C. Vogel, K. Schindler, S. Roth, 3D scene flow estimation with a rigid motion prior, in: International Conference on Computer Vision, 2011.

[21] L. Valgaerts, A. Bruhn, H. Zimmer, J. Weickert, C. Stoll, S. Theobalt, Joint estimation of motion, structure and geometry from stereo sequences, in: European Conference on Computer Vision, 2010.

[22] J. Neumann, Y. Aloimonos, Spatio-temporal stereo using multi-resolution division surfaces, International Journal of Computer Vision (2002) 181–193.

[23] Y. Fukurawa, J. Ponce, Dense 3d motion capture from synchronized video streams, in: Conference on Computer Vision and Pattern Recognition, 2008.

[24] R. Carceroni, K. Kutulakos, Multi-view scene captured by surfer sampling: From video streams to non-rigid 3D motion, International Journal of Computer Vision 110 (2008) 75–90.

[25] F. Devernay, D. Mateus, M. Guilbert, Multi-camera scene flow by tracking 3D points and surfels, in: Conference on Computer Vision and Pattern Recognition, 2006.

[26] H. Spies, B. Jahne, J. Barron, Dense range flow from depth and intensity data, in: International Conference on Pattern Recognition, 2000.

[27] T. Lukins, R. Fisher, Colour constrained 4d flow, in: British Machine Vision Conference, 2005.

[28] J. Fang, T. Huang, Solving three-dimensional small-rotation motion equations, in: Conference on Computer Vision and Pattern Recognition, 1983.

[29] H. Longuet-Higgins, K. Prazdny, The interpretation of a moving retinal image, Royal Society London 208 (1980) 385–397.

[30] B. Horn, E. Weldon, Direct methods for recovering motion, International Journal of Computer Vision 2 (1988) 51–76.

[31] G. Adiv, Determining three-dimensional motion and structure from optical flow generated by several moving objects, IEEE Transactions on Pattern Analysis and Machine Intelligence 7 (1985) 384– 401.

[32] J. Bergen, P. Anandan, K. Hanna, R. Hingorani, Hierarchical model-based motion estimation, in: European Conference on Computer Vision, 1992.

[33] B. Lucas, T. Kanade, An iterative image registration technique with an application to stereo vision, in: International Joint Conference on Artificial Intelligence, 1981.

[34] T. Brox, A. Bruhn, N. Papenberg, J. Weickert, High accuracy optical flow estimation based on a theory for warping, in: European Conference on Computer Vision, 2004.

[35] P. Green, Iteratively reweighted least squares for maximum likelihood estimation, and some robust and resistant alternatives, Journal of the Royal Statistical Society 46 (1984) 149–192.

[36] J. Shi, C. Tomasi, Good features to track, in: Conference on Computer Vision and Pattern Recognition, 1994, 1994.

[37] D. Scharstein, R. Szeliski, High-accuracy stereo depth maps using structured light, in: Conference on Computer Vision and Pattern Recognition, 2003.

[38] J. Bouguet, Pyramidal implementation of the Lucas-Kanade feature tracker, Open CV Documentation, Technical Report, Intel Corporation, Microporcessor Research Labs, 1999.

[39] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. Black, R. Szeliski, A database and evaluation methodology for optical flow, in: International Conference on Computer Vision, 2007.

[40] D. Herrera, J. Kannala, J. Heikkila, Accurate and practical calibration of a depth and color camera pair, in: International Conference on Computer Analysis of Images and Patterns, 2011.

[41] J. Quiroga, F. Devernay, J. Crowley, Scene flow by tracking in intensity and depth data, in: Conference on Computer Vision and Pattern Recognition Workshops, 2012.